2005

CluSID: a clustering scheme for intrusion detection, improved by information theory

R. Shokri
University of Tehran, Iran

Farhad Oroumchian
University of Wollongong in Dubai, farhado@uow.edu.au

N. Yazdani
University of Tehran, Iran

CluSID: a Clustering Scheme for Intrusion Detection, Improved by Information Theory

Reza Shokri
ECE Department
University of Tehran
Tehran, Iran
r.shokri@ece.ut.ac.ir

Farhad Oroumchian
College of Information Technology
University of Wollongong in Dubai
Dubai, UAE
foroumchian@acm.org

Nasser Yazdani
ECE Department
University of Tehran
Tehran, Iran
yazdani@ut.ac.ir

Abstract—**Security is a big issue for all networks in any enterprise environment. Many solutions have been proposed to secure the network infrastructure and communication over the Internet. Intrusion Detection Systems with many different techniques such as data mining approaches are employed to maximize the detection rate of intrusions while reducing false alarm rate. For instance, many clustering techniques are recommended which segregate normal and abnormal data in IDSs. Clustering methods put emphasis on finding differences and similarities of traffic sessions to categorize each one in its corresponding groups. These groups are represented by their assigned labels. Later, these labels are used to predict the type of the incoming network traffic. In this paper, we propose a clustering scheme to use in intrusion detection systems, named CluSID. The major contribution of CluSID is using information theory for taking full advantages of clustering techniques. The main logic behind CluSID is to use non-uniform gain functions for network traffic features in order to improve the accuracy of clustering process. To this end, we apply information theory concepts for moving center of clusters to the most important areas in the domain of the selected features. The results clearly show a raise in detection rate of CluSID in most of the attack categories in comparison to KDD CUP'99 Winner and simple clustering methods. The increase in detection rate of proposed system is about 25 percent.**

Keywords- Intrusion Detection System, Clustering, Information Theory, Entropy.

I. INTRODUCTION

Intrusion detection is the process of monitoring computer networks and systems for recognition of security policy violations. Intrusion detection augments the traditional audit, which was designed to occur at infrequent intervals, thus, making it a continuous process [10].

Anomaly and misuse detection are two major areas of research in Intrusion Detection Systems. Misuse detection systems encode and match the sequence of "signature actions" of known intrusion scenarios. The main shortcoming of such systems is that known intrusion patterns have to be hand-coded into the system; they are unable to detect any future (unknown) intrusions that have no matched patterns stored in the system. *Anomaly detection* systems establish normal usage patterns (profiles) using statistical measures on system features, i.e. syscalls of a particular user or program. The main weakness of these systems is that intuition and experience is relied upon in selecting the system features, which can vary enormously among different computing environments; some intrusions can only be detected by studying the sequential interrelation between events because each event alone may fit the profiles. Therefore, the false alarm rates could be high.

An intrusion Detection System consists of an audit data collection agent that collects information about the system being observed. This data then is either stored or processed directly by the detection engine. The output of the detection engine is presented to the response system or the site security administrator, who then can take further actions, normally beginning with further investigation into the causes of the alarm.

Wide variety of techniques have been applied to both misuse and anomaly detection. Artificial Intelligence (AI) and Rule based expert systems have served as the basis for several systems. The acquisition of expert system rules is a tedious and error-prone process. This problem has generated a great deal of interest in the application of machine learning techniques to automate the process of pattern learning. Examples include the Time-based Inductive Machine (TIM) for intrusion detection that learns sequential patterns and neural network-based intrusion detection systems. More recently, techniques from the fuzzy computing area have been used to separate normal patterns from audit data and solve the problem of uncertainty in defining the necessary thresholds.

As another solution, various data mining algorithms drawn from fields of statistics, pattern recognition, machine learning and database system have been deployed in the technology of intrusion detection. Here are a few specific things that data mining might contribute to an intrusion detection project:

1) Remove normal activities from alarm data to allow analysts to focus on real attacks.
2) Identify false alarm generators and bad sensor signatures.
3) Find anomalous activities that uncover real attacks.
4) Identify long, ongoing patterns.
5) Determines relations between various fields in a database, and in data stream.

To accomplish these tasks, data miners use one or more of the following techniques:

1) *Data summarization* with statistics.
2) *Visualization*: presenting a graphical view of the data.
3) *Clustering*: putting data into natural categories [6].

1-4244-0000-7/05/$20.00 ©2005 IEEE.

4) Association rule discovery: defining normal activity and enabling the discovery of anomalies ([7], [8]).

5) Classification: classifies a data item into one of several pre-defined categories [9].

Clustering is the method of grouping objects into meaningful subclasses so that members from the same cluster are quite similar and members from different clusters are quite different from each other. Therefore, clustering methods can be useful for classifying log data and detecting intrusions.

The main contribution of this paper is improving the detection rate of a Clustering Scheme for Intrusion Detection (CluSID), using Information Theory. Constructing well structured clusters makes the intrusion detection simpler in CluSID. Moreover, factor analysis is being done using Information Theory that could significantly help us to extract important features. CluSID detects %83.48 of intrusions. It is worth noting that this is an improvement with respect to KDD CUP'99 Winner and systems based on simple clustering. Furthermore, the probing attacks are detected with a considerably higher rate.

The next section of this paper surveys the related work in Data Mining frameworks and algorithms used in intrusion detection. Section three depicts the design structure of proposed system, and explains each part of the CluSID. Section four reports the experimental results, and finally, section five concludes the paper.

II. Related Work

Use of different classification and clustering algorithms for Intrusion Detection has been reported in literature. Furthermore, several Data Mining techniques have been adapted for generating rules to find out normal and abnormal signatures in network traffic.

Using data mining algorithms for classification of network traffic and associating rules to them for network intrusion detection have been proposed in [14] and [16]. Also, in [15], a data mining framework for constructing intrusion detection models is outlined. The key idea is to apply data mining programs on audit data to compute misuse and anomaly detection models, according to the observed behavior in the data. Meta-learning has also been proposed as a means of constructing a combined model that incorporate evidence from multiple base models. In addition, the basic association rules and frequent episodes algorithms have been extended to consider the special requirements in analyzing audit data.

Furthermore, classification algorithms play important roles in intrusion detection systems. Most of these algorithms developed until now somehow try to measure the relative importance of an attribute in classification compared to others and use this knowledge while classifying objects. Some of them (i.e. [1]) try to determine the relevant and irrelevant features in a set of attributes in order to take relevant features into account more than irrelevant ones. Others (i.e. [2]) attempt to assign weights according to their degree of relevance on classification of instances. In the later methods, some transformation based approaches and genetic algorithms are used to determine the weight of each attribute in a dataset [3]. A classification algorithm called ID3, which introduces the concept of information gain, is proposed in [4], [5].

Moreover, K-means [13] is a typical clustering algorithm. It partitions a set of data into k clusters through several steps. K-means has been used for clustering data for decades. However, it has two shortcomings in clustering large data sets: number of clusters dependency and degeneracy. Number of clusters dependency is that the value of k is very critical to the clustering result. Obtaining the optimal k for a given data set is an NP-hard problem [12]. Degeneracy means that the clustering may end with some empty clusters. This is not what we expect since the classes of the empty clusters are meaningless for the classification. The H-means+ algorithm, proposed in [12], can overcome the weakness by replacing the empty cluster with a newly created cluster. The center of the new cluster is the global furthest point of the data set. The global furthest point is the local furthest point with the greatest distance from its local center; and the local furthest point of a cluster is the remotest point from the cluster center. When an empty cluster is found in an iteration of K-means, the global furthest point will be removed from its cluster and be designated as a new cluster center to replace the empty cluster center. After that, K-means iteration will go on until no empty cluster exists.

III. CluSID Design Structure

Figure 1 illustrates the design of CluSID system's training and testing phases. In the training phase, the KDD training data is clustered based on the result of Feature Selection and Gain Computation modules. The centroid of each cluster is also calculated and passed to the Detection Engine. Then, CluSID engine uses these centroids to classify the incoming sessions from the KDD test data.

DARAPA has provided a standard set of data which includes a wide variety of intrusions simulated in a military network environment. The 1999 KDD intrusion detection contest used a version of this dataset [11].

The raw training data was about four gigabytes of compressed binary TCP dump data. This was processed into about five million connection records. Similarly, the test data yielded around two million connection records. Each connection is labeled as either normal or as an attack, with exactly one specific attack type. Attacks fall into four main categories:

1) DoS: Denial of Service, e.g. syn flood.

2) R2L: Unauthorized access from a remote machine, e.g. guessing password.

3) U2R: Unauthorized access to local superuser (root) privileges, e.g., various "buffer overflow" attacks.

4) Probing: Surveillance and other probing, e.g., port scanning.

In the *Training Phase*, we have deployed several components. As shown in Figure 1 we have used KDD train data to select features and extract the centers of eight clusters. These eight clusters consist of four attack and four normal clusters. Number of clusters, are selected based on experimental results, as explained in section IV.

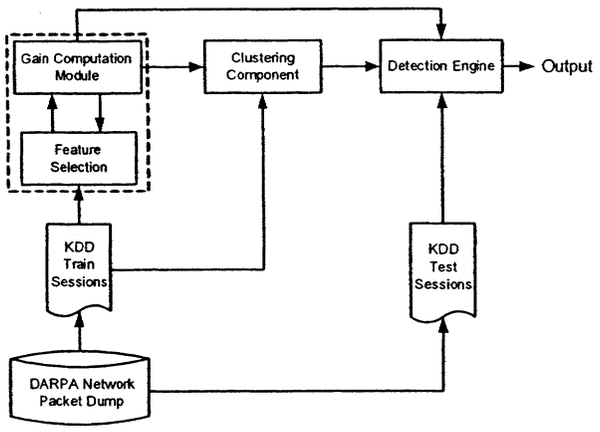


Figure 1. CluSID design structure.

Every KDD connection record consists of 41 features. So, the task of *Feature Selection* component and *Gain Computation Module* is to select the more valuable features. In this way, a subset of these 41 features is selected to detect the attack sessions. Details of the methods that have been used to accomplish this task will come in the next sections.

As mentioned above, every KDD training session has a label which is either "normal" or an attack category name. The *Clustering* component uses this information to cluster the training data and compute the centers of the eight clusters. After computation of the centers, mentioned component uses a refinement method to improve its results. As the result of training phase, center of eight clusters with selected features are calculated, considering their gain value.

In the *Testing Phase*, the results of training phase and the KDD test data are used. The detection engine labels every session to be an attack type (DoS, R2L, U2R, and Probe) or Normal. The result of this labeling is compared with correct labels provided by KDD to analyze the system.

A. Computation of Pure Feature's Gain

Different weights are assigned to different features for representing the significance of them in traffic sessions. The Gain function, first estimates these weights, by means of calculating the pure gain of features. However using a uniform weight for the values of the domain of a feature might reduce the effect of more important values in favor of less important values. Therefore pure gain values are not sufficient for fine detections. But, in this section we use Information Theory concepts to compute the gain, as is called; Pure Gain values, in order to use and refine them in Gain Functions.

The set of values that could be assigned to a feature represents the domain of that feature. Feature's domains could be continuous or discrete. For example, {tcp, udp, icmp} are values in the domain of "protocol_type" feature, and also the domain of "srv_serror_rate" feature that shows the percentage of connections having "SYN" errors, is Real numbers between 0-100.

Let S be the set of n instances and let C be the set of k domains. Let $P(C_i, S)$ be the fraction of the examples in S that have domain C_i. Then, the expected information from this domain membership is as follows:

$$Info(S) = -\sum_{i=1}^{k} P(C_i, S) \times \log(P(C_i, S)) \quad (1)$$

Also, the Entropy of attribute A, that is spitted into subsets $\{A_1, \ldots, A_v\}$ is:

$$Entorpy(A) = -\sum_{i=1}^{v} \frac{|A_i|}{|S|} \times Info(A_i) \quad (2)$$

Where $|S|$ is the number of instances in S, and $|A_i|$ is the number of instances of A belongs to A_i.

The Entropy is a measure of how much uncertainty is involved in the selection of a symbol – the greatest the entropy, the greatest the uncertainty. It can also be considered a measure of the "information content" of the feature – more probable attribute conveys less information than less probable ones.

Then, the difference between *Info(S)* and *Entropy(A)* gives the information gained by partitioning S according to testing A that we are named Pure Gain.

$$PureGain(A) = Info(S) - Entropy(A) \quad (3)$$

This measure shows the value of an attribute in contrast to all the other attributes. So a feature with high Pure Gain has more significance in the decision mechanism.

B. Computation of Feature's Gain Functions

Calculating the information gain is straightforward for nominal attributes since there are a finite number of distinct values of each nominal attributes, and each instance is associated with one of those values. Because most of the features extracted in network sessions have continuous values, the Pure Gain, formulated in (3), is not suitable as weight in our clustering algorithm. Also, there are some ranges of values in the domain of features which are more important for detecting an attack. For example many attacks have used 0-200 bytes of data for running their scenarios. Therefore a value between 0 and 200 for "Number of Bytes of Data from Destination to Source" feature is more important than any other value outside this range. Therefore, using a uniform Pure Gain value for domain of a feature will decrease the detection power, and increase the rate of false alarm in Intrusion Detection System.

Subsequently, we propose a partitioning algorithm for distinguishing among the ranges of values in a domain, based on their significance for detecting an attack. Gain of a feature is a function that indicates usefulness of such feature in throughout of its domain. After calculating pure gain for each feature, a partitioning algorithm estimates the Feature Gain functions, based on a statistical routine. Figure 2 shows the partitioning result of sent data for about 200 – for cleanness – sessions of KDD CUP'99 data. Distance of each point from center, shows its value.

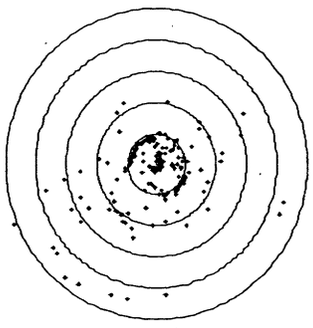


Figure 2. Partitioning of sample data, based on its feature values.

Partitioning algorithm in a domain is the process of splitting a feature's domain to find more important and sensitive sub-domains within a domain. For finding more important sub-domains, we divide the domains into sub-domains and count the number of samples (by means of Count function) in each sub-domain or partition. Partitions with high count values are assumed more important. The process starts with a constant value as the number of partitions and iterates splitting partitions until the difference of *MaxCount*, as the maximum value of Count Function in all partitions, and 2^{nd}*MaxCount*, as the second greatest value of the Count function, reaches to a pre-learned threshold. By using normalized Count and selecting the threshold based on it (difference of *MaxNormalCount* and 2^{nd}*MaxNormalCount*), same threshold could be used for all features. Wise selection of the threshold can help the algorithm to avoid eliminating smaller values by comparing them to higher values. Furthermore, based on experimental results, we have selected 0.34 as the optimum value for this threshold. However, this method is not suitable for discrete domains such as "protocol_type" and "service_type". For the features with discrete domain, a good choice for the number of partitions is the number of discrete values in the domain of the feature. (i.e. "protocol_type" will be partitioned to 3 parts: tcp, udp, and icmp)

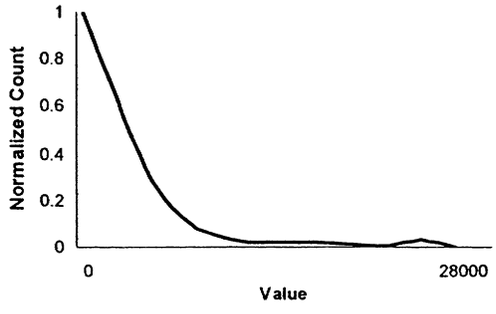


Figure 3. Normalized Count vs. corresponding Value of a feature.

After finishing the iterations, final Gain of the features can be computed by multiplying the normalized count and pure gain, as formulated in (4) and illustrated in Figure 3.

Let x be the value of feature A in partition i, so the gain function will be:

$$Gain(x) = \frac{Count(i)}{MaxCount} \times PureGain(A) \qquad (4)$$

C. Feature Extraction

A primary problem in Intrusion Detection Systems is identifying a representative set of features from which to construct a classification model for a particular manner (normal, abnormal). The central hypothesis is that good feature sets contain features that are highly correlated with the data specifications, yet uncorrelated with each other.

In CluSID, feature selection is done by eliminating features with pure gain value less than a threshold. Setting the threshold value is a trade-off between the power of detection rate and speed of the detection. Since not all the features have similar gain values, elimination of features with low gain values will have minimal effect on the detection rate.

D. Clustering of Sessions

Clustering of traffic sessions and partitioning data into clusters can help to identify specification of normal and abnormal data and differentiate their patterns. Current clustering methods such as k-means have been used for such purposes. Using these techniques without considering the behavior of network traffic may cause many false alarms and/or miss errors.

In CluSID the task of clustering component is to create the clusters, find their centers, and pass this information to Detection Engine. Based on the labeling of attacks in DARPA data set, abnormal sessions are organized in clusters with associated attacks category labels. In addition, based on our experiments, it seems better to cluster normal data into four clusters (equals to the number of abnormal data clusters). It seems representing normal data with a single cluster, leads to place the centroid of normal data in a position where the chance of mislabeling marginal data (that are located in the borders of abnormal data) is increased. Clustering of training sessions includes two major steps:

- Primary clustering.
- Refinement of clusters.

As mentioned before, primary clustering of abnormal data is only based on the DARPA labels. But for normal data, a refined k-means clustering algorithm is used to cluster normal data by considering their gain functions.

The procedure follows a straightforward way to classify a given data set through a certain k number of clusters (4 clusters in normal data set) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because the location of the centroids affects the results. So, the best choice is to place them as much as possible far away from each other. The next step is to take each member of the training data set and associate it with the nearest centroid, based on the distance formula. When no point is left, the first step is completed and an early grouping is done. At this point we need to re-calculate k new centroids based on the features weight of cluster centers, resulting from the

previous step. After having these k new centroids, a new binding has to be done between all of the data set points and the nearest new centroid. This process is repeated again and in each iteration all centroids change their location step by step until no more changes are possible. In other words, centroids do not move any more.

Above algorithm aims at minimizing an *objective function*, in this case, weighted squared error function. The objective function is:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} D_{ij} \quad (5)$$

Where, k and n are the number of clusters and training dataset members, respectively, and D_{ij} is the distance of i^{th} data point from j^{th} cluster center.

Let f be the number of selected features and G_l is the return value of gain function for feature l, so D_{ij} is computed as following:

$$D_{ij} = \sum_{l=1}^{f} G_l \left| x_{il} - c_{jl} \right|^2 \quad (6)$$

Where x_{il} is value of l^{th} feature for point i, and c_{jl} is value of l^{th} feature for center of j^{th} cluster.

Primary clustering for all data has finished as described above. Another round of k-means clustering is performed for the eight clusters to tune the centers and reduce the error rate in detection. Based on our experiments, a constraint of "*boundary distance*" should be applied to the points to prevent their migrating from a cluster to another cluster, if their distance from associated clusters is less than boundary distance. Half of the minimum distance between cluster's centers is a good value for boundary distance.

E. Detection Method

The Detection Engine's mission is to find appropriate clusters for the test data. For these test data, CluSID calculates each data point's position in the cluster space. Detection engine has the specification of the features and their gain functions, plus the center of the clusters.

```
Min = Dii;
MinID = 1;
For j= 2 to 8 {
        Calculate Dij;
        If (Dij < Min) then {
                Min = Dij;
                MinID = j;
        }
}
```

Figure 4. Pseudocode for finding corresponding cluster for x_i.

Therefore, base on (6) it calculates which cluster the sample test session belongs to. Figure 4 illustrates the process of finding corresponding cluster for session x_i. The *MinID* is the cluster number that the session x_i belongs to. Based on the most similar cluster, CluSID can label the session as normal or abnormal.

IV. Experimental Results

About 600,000 sessions of DARPA Data Set has been selected and used for training phase of our system. Furthermore about 400,000 sessions have been used for testing the system. Statistical analysis and computation of gain values are done by Matlab. Partitioning the domain of features was required to calculate gain values. We have selected 0.34 as the optimum value for the above mentioned threshold that is used by partition algorithm to stop partitioning. This value is learned from simulations, and calculating corresponding detection rate for different threshold values.

After normalizing the gain for each feature and calculating gain functions, 13 features with smallest values have been eliminated. Elimination of more features suddenly decreases the power of detection engine. Although the results show high detection power for CluSID, but using all the features (no feature reduction) actually exhibits even higher detection rate in the expense of more time spent on detection (more useful in offline detection systems).

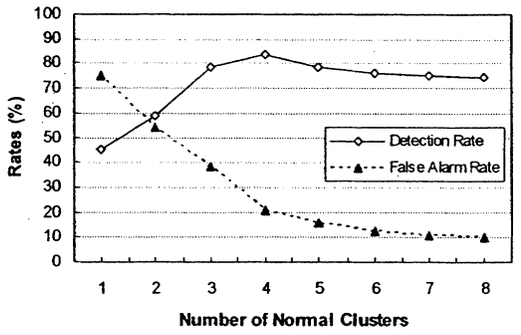


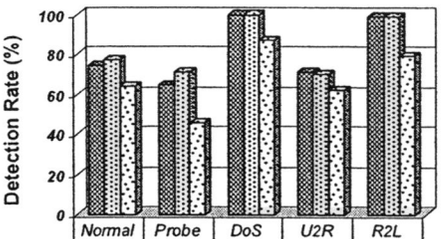
Figure 5. Detection rate of CluSID vs. number of normal clusters.

As mentioned before, four clusters have been created that specify abnormal groups in DARPA data set. We have investigated changing effect of normal cluster numbers from one to eight on the system's detection rate. Figure 5 depicts the result of this test. As it is evident in the diagram by considering the detection rate and the false alarm rate, the system performs better when the number of normal clusters is equal to the number of abnormal clusters. Using fewer numbers of normal clusters leads to setting the center of normal clusters between the abnormal data clusters, consequently increasing the false alarm rate of the system. As shown in the graph, raising the number of normal clusters results in reducing the number of False Alarms as well as the detection rates. Because the detection rate of CluSID is more important for us, we choose four clusters as normal clusters.

After clustering and refinements phases, eight points are discovered, as center of clusters. So the CluSID detection

engine is constructed with 28 gain functions and 8 cluster centroid points.

Finally, we compare the detection rate of CluSID against that of the KDD CUP'99 Winner, and a detection engine based on simple clustering. As depicted in Figure 6 in almost all categories, the CluSID has demonstrated equal or better detection rate. Furthermore, Normal data and Probing attacks can be detected more accurately by CluSID.



	Normal	Probe	DoS	U2R	R2L
KDD CUP'99 Winner	74.6	64.8	99.9	71.4	98.8
CluSID	77.2	71.2	99.9	70.2	98.9
Simple Clustering	64	46	87	62	79

Figure 6. CluSID detection rate vs. KDD CUP'99 winner, and a simple clustering system.

CluSID's ability to detect some attacks better than other systems is contributed to its method of partitioning the domain of the features. Also, some attacks that behave like normal traffic can be discovered by CluSID. As shown in the diagram, our scheme improves the detection rate by 25% on average, in comparison with the simple clustering systems. In addition, the CluSID detects probe attacks more accurately (improvement of about 10 percent) in contrast to KDD CUP'99 Winner.

Using all the 31 features of traffic sessions to construct the detection engine, plus using all the training sessions of DARPA Data Set, could help the system to produce even better results.

V. Conclusion

In this paper, we have outlined a clustering method for constructing a detection engine named CluSID for intrusion detection systems. Information theory concepts have been employed for refining the center of clusters which result in a better classification of incoming traffic sessions. An efficient feature selection method is presented to choose the most appropriate features by setting a threshold on gain values. On the other hand, the Cross-Entropy techniques can be utilized for this purpose.

Our experimental results show that the detection rate achieved by the CluSID system is 25 percent higher than simple clustering methods. Also, CluSID outperforms KDD CUP 99 Winner in most attack categories. For example, CluSID has about 10 percent improvement in probe attack detection.

Since CluSID is flexible to tradeoff between high detection rate and high speed detection, our proposed method might be applied on various systems. Also, small amount of memory that is used in the detection engine, make CluSID more attractive to be used on IDS systems with restrict resources such as wireless sensor networks.

Acknowledgment

We wish to thank our colleagues at University of Tehran, Router Lab, especially Mr. Nayyeri and Mr. Zarifzadeh, and also Mr. Varshovi from Data Security Laboratory of Amirkabir University of Technology, for their help and encouragement.

References

[1] R. Kohavi, G. H. John, and K. Pfleger, "Irrelevant features and the subset selection problem". In Proceedings of the 11th Intl. Conf. on Machine Learning (ICML '94), pages 121-129, 1994.

[2] P. Langley and A. L. Blum, "Selection of relevant features and examples in machine learning", Special Issue of Artificial Intelligence on Relevance, 1994.

[3] J. D. Kelly and L. Davis, "A hybrid genetic algorithm for classification", In Proceedings of the 12th International Joint Conference on Artificial Intelligence, pages 645-650, 1991.

[4] J. R. Quinlan, "Induction of decision trees", Machine Learning, 1, 1986.

[5] J. R. Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann, California, 1993.

[6] S. Manganaris, M. Christensen, D. Zerkle, and K. Hermiz, "A data mining analysis of RTID alarms", Computer Networks, 34, p. 571-577, 2000.

[7] C. Clifton, and G. Gengo, "Developing Custom Intrusion Detection Filters Using Data Mining", 2000 Military Communications International, Los Angeles, California, October 22-25, 2000.

[8] D. Barbara, N. Wu, and S. Jajodia, "Detecting Novel Network Intrusions Using Bayes Estimators", Proceedings Of the First SIAM Int. Conference on Data Mining, (SDM 2001), Chicago, IL, 2001.

[9] W. Lee, and S. Stolfo, "Data Mining Approaches for Intrusion Detection", in Proceedings of the 7th USENIX Security Symposium, San Antonio, TX, 1998.

[10] RG Bace, "Intrusion Detection". MacMillan Technical Publishing, 2000.

[11] KDD Cup 1999 Data. University of California, Irvine, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999.

[12] P. Hansen and N. Mladenovic. "J-means: a new local search heuristic for minimum sum-of-squares clustering", Pattern Recognition, 34(2):405.413, 2002.

[13] J. MacQueen. "Some methods for classification and analysis of multivariate observations". Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability, 2:28. 297, 1967.

[14] W. Lee, S. J. Stolfo, and K. Mok, "Mining in a data flow environment: Experience in intrusion detection", In Proceeding of the 1999 Conference on Knowledge Discovery and Data Mining KDD-99.

[15] W. Lee, S. J. Stolfo, and K. Mok, "A data mining framework for building intrusion detection models", In Proceedings of the 1999 IEEE Symposium on Security and Privacy, May 1999.

[16] W. Lee, S. J. Stolfo, and K. Mok, "Data Mining Approaches for Intrusion Detection", In Proceedings of the 7th USENIX Security Symposium, 1998.

[17] Robert M. Gray, "Entropy and Information Theory", by Springer Verlag, 1990.

[18] David J.C. MacKay, "Information Theory, Inference, and Learning Algorithms", Cambridge University Press, 2003.

[19] P. Berkhin. "Survey of Clustering Data Mining Techniques", Accrue Software Inc.